UCRL-AR-121011
May-June 1995

Computation/AIS

Software Engineering Newsletter

Published by the Software Engineering Working Group and the Software Technology Center to promote software engineering education

Upcoming Events

TakeFive Demo

On June 21, in the south cafeteria: TakeFive Software will demonstrate SNiFF+, a programming environment for Unix C and C++ development. SNiFF+ incorporates software engineering capabilities such as impact analysis, virtual workspace support and automated documentation to create a software development environment.

SNiFF+ includes a comprehensive toolset providing team support, code comprehension and reverse engineering, edit-compile-debug and documentation support, build management and tool integration.

Together these tools are desinged to improve developer productivity, promote software reuse and enhance software quality for even large projects working with very complex software systems.

Metrics Seminar

On August 19th, in the B439 Training Room; Speakers will be Terri Quinn, and Steve Wong.

More details to come...

Seminar on "Systems Engineering Capability Model"

Coming later this summer, Roger Bate of SEI will be here for this seminar. More details to come...

Personal Software Process Class

CTEC Send out a flier for this class it will be starting on August 11. Please read the flier or call Christa Sobczak X2-4257 or the STC for more details.

In this issue:

C+MPP

1995 Symposium

LLNL Symposium on Distributed Computing and Massively Parallel Processing, B123 Auditorium, June 7-9, 1995

For more info e-mail Jennifer Gibson at stc@llnl.gov

Insure++, A C Programmer's Prospective

Rob Neely, rneely@llnl.gov
L-035, X 3-4243

For a C programmer, there are two essential items which, in my opinion, should be readily at hand during a project: K&R's second edition, and Insure++ from Parasoft.

One of the most immediately invaluable tools in the Insure++ suite is called "Insight." Insight helps right many of the wrongs with the C language - namely the ability to tromp through memory at will, giving no indication of anything being wrong. Insight claims to catch:

+ Memory corruption due to reading or writing beyond valid areas of global, local, shared, or dynamically allocated object;
+ Operations of illegal, or unrelated pointers;
+ Reading uninitialized memory;
+ Memory leaks;
+ Errors allocating and freeing dynamic memory;
+ Some other things, which judicious use of an ANSI C compiler will also catch.

I will personally testify that these claims are entirely valid. Insight works by instrumenting the source code with calls to all sorts of assertions. Other competing products only work on object files, and thus aren't quite as good at catching every last memory error. I know - I've used them too.

I have been using Insight now for about 2 or 3 months on a medium scale project - about 15,000 lines of code and growing. Regardless of the fact that my C code compiled cleanly under a C++ compiler, I was rather blown away at the number of tiny (and not so tiny) errors my program had the first time I ran it through Insight, even though it appeared to run correctly on my workstation.

Since then, I have been using Insight as a first line of defense in tracking down bugs that creep into my program during a blast of new code additions. Not only will it help you keep your code cleaner by finding things like memory leaks, but can also help you immediately pinpoint bugs caused by things such as out of bounds array references - even non-unit-strided access to dynamically allocated arrays.

The downside to using Insight is the amount of time it adds to the compile and execution of your program. Compiles generally take about 5-7 times longer than your standard C compiler without optimization, and the increase in execution time is on the same order. However, I have found that this extra time spent more than pays off (by a long shot) in the long run, as Insight virtually eliminates the need to spend days tracking down a memory error - which we all know is very frustrating.

In addition to the basic Insight tool described above, there are several things bundled with the Insure++ package which are quite helpful during the software life cycle. "InUse" allows you to interactively view at run time certain features of your program such as memory usage, heap layout, block sizes of mallocs, etc... "Invision" lets you view memory access patterns for a particular piece of code, and can thus give you help in optimizing your algorithms. Both of these tools are graphically based. The TCA (Total Coverage Analysis) tool is especially useful during the testing phase. It keeps track of each line of code which has been executed by your program over the course of its lifetime, thus allowing you to create input cases to test branches of your code which have not yet ever been executed.

In my opinion, nobody should program C code without using some sort of system for finding possible memory errors. Parasoft has invented an excellent integrated system for doing this. For more details, check out http://www.parasoft.com/insure.html.

Software Architecture

Al Leibee, leibee1@llnl.gov,
L-307, X 2-1665

The field of software architecture is an area of active research in both industry and academia. This year's Software Technology Conference, with an attendance of 2800, had an entire track devoted to software architecture. There is not yet a consensus on the definition of software architecture, but there is general agreement that software architecture is both a discipline of design and a representation of design and identifies the following software attributes:

- Computational/functional and data components
- Connections between components, including data flow and control flow
- Constraints, including communication protocols, visibility, timing, and synchrony
- Topological notion of the structure formed by components and their connections

There are architectural styles such as distributed, layered, and client/server. Examples of architectural constraints are throughput and timing. A database management system can be an architectural component. Architectural connectors include procedure calls and pipes.

Garlan and Shaw's "An Introduction to Software Architecture" (SEI-94-TR-02) lists the following areas of study for software architecture:

- Taxonomies of architectures and architectural styles
- Formal models for characterizing and analyzing architectures
- Notations for describing architectural designs
- Tools and environments for developing architectural designs
- Techniques for extracting architectural information from existing code
- Better understanding of the role of architectures in the life-cycle process

UNISYS has put a Software Architecture Technology Guide on the World-Wide Web at http://www.stars.reston.unisysgsg.com/arch/guide.html. The guide describes many of the concepts of software architecture and has a bibliography of books, papers, and articles on the subject.

Metrics Food For Thought and Facts

Al Leibee, leibee1@llnl.gov,
L-307, X 2-1665

From the newsletter "IT Metrics Strategies", edited by Howard Rubin, on the topic of metrics visualization—

"An excellent area for using these concepts (of communicating information via graphical displays of information), and a prime candidate for "metrics visualization" is the area of organizational readiness. Information technology organizations must constantly face change. Change may come in the form of a new technology, a new methodology, a new process discipline, and even a new business environment. A core competency for today's IT organization is the ability to manage and navigate change. To do so requires an understanding of the dimensionality of the required change and the ability to chart a course to make it happen.

One way of doing this is through metrics visualization. An organization must be able to characterize its "as-is" state (where it is today) and the attributes of it's "to-be" state. ..."

Also from "IT Metrics Strategies", Howard Rubin's 1994 industry survey shows 47% of development effort is spent on maintenance (corrective, adaptive, and perfective activities) and 53% on new development. In the software producing industry, 46% is spent on maintenance, 54% on new development. The survey data was gathered from attendees

Continued on page 7...

Upcoming Seminars and Conferences

June

19-20 Software Configuration Management; 2 day Seminar
Hyatt San Jose, California
Info or other course listings: (201) 478-5400

27-30 25th International Symposium on Fault-Tolerant Computing
Pasadena, California
Info: anonymous FTP, ftp.cs.ucla.edu: /pub/ftcs25

July

12-16 13th International System Safety Conference
Red Lion Inn San Jose, California
Contact: Michael Scannell (408) 742-9581
or mscannell@lmsc.lockheed.com

14-16 2nd Working Conference on Reverse Engineering
Toronto, Ontario, Canada
in conjunction with CASE'95
Info: Hausi Muller, hausi@csr.uvic.ca

Sept

27-29 13th Annual Pacific Northwest Software Quality Conference
Portland, Oregon Convention Center
Contact: Terri Moore (503) 223-8633

Nov

6-10 1st International Conference on Engineering of Complex Computer Systems
Southern Florida
Info: Alexander Stoyenko, alex@vulcon.njit.edu

The following are being offered by the ***Software Engineering Institute****. For more info:*

Internet: registration@sei.cmu.edu or Phone: 412 / 268-7388

June	*20-22*	*Defining Software Processes*
	28-29	*Annual Disciplined Engineering Workshop: Effective Practice in Performance Engineering*
August	*15-17*	*Engineering an Effective Software Measurement Program*
Sept	*11-14*	*SEI Software Engineering Symposium*
	18-22	*Consulting Skills Workshop*

Local Lab experts offer advice, expertise

(most of these people belong to the Software Engineering Working Group, SEWG)

Reviews and Walkthroughs
- Carmen Parrish
- Warren Persons, 2-3349
- Jeff Young
- Carolyn Owens

JAD/FIND
- Candy Wolfe

If you need consulting help with a project involving software engineering, consider contacting one of the local LLNL experts.

Performance, Reliability & Safety
- Dennis Lawrence

Reverse Engineering
- Jeff Young
- Al Leibee

Requirements Modeling/OOD
- Debbie Sparkman

Testing
- Warren Persons, 2-3349
- Nancy Storch
- Al Leibee

Software Quality Assurance
- Warren Persons, 2-3349

CASE Tools
- Suzanne Pawlowski
- Jeff Young

Configuration Management
- Al Leibee
- Carmen Parrish

Project Estimation/Management
- Howard Guyer, 3-7671
- Carolyn Owens

SEWG Members:

- Bill Aimonetti, 3-2678
- Bill Buckley, 3-4581,
- Bob Corey, 3-3271
- Antonia Garcia, 3-9884
- Howard Guyer, 3-7671
- Al Leibee, 2-1665
- Judith Littleton, 3-4403
- Donna Nowell, 2-1515
- Jerry Owens, 2-1646
- Carolyn Owens, 3-6085
- Carmen Parrish, 2-9810
- Suzanne Pawlowski, 3-0115
- Frank Ploof, 2-6990
- Terri Quinn, 3-2385
- Denise Sumikawa, 2-1831
- John Tannahill, 3-3514
- Booker Thomas, 3-8800
- Ernie Vosti, 3-0604
- Jeff Young, 3-8333
- Bill Warren 2-5331
- Candy Wolfe, 2-1863

SEWG Meetings are held every 1st and 3rd Thursday 3:00 to 4:00, in B218, Room 114.

Focus On Metrics: Part 3

Al Leibee, leibee1@llnl.gov,
L-307, X 2-1665

In the previous issue, I described a top-down approach for determining the metrics to be collected. This was the Goal/Question/Attribute/Metric method that starts by defining high-level business goals and then derives from these goals the metrics needed to support them. The bottom-up approach starts with measurable observations and then builds up management objectives and goals. The bottom-up method I'll describe in this article was developed by Bill Hetzel, author of the book "Making Software Measurement Work", and by Bill Silver. Their method focuses on the work products of the software development process. Examples of work products are design specifications, source code, and test cases. The eventual work product is the system used by the customer. Their method, the IOR method, specifies three categories of metrics to be defined for each work product—-

1. Input metrics
Metrics that quantify the resources, activities, and other work products used in the creation of the work product.

2. Output metrics
Metrics that quantify the work product itself such as a size metric.

3. Results metrics
Metrics that quantify the usefulness and quality of the work product.

For example, suppose the work product is the source code module. The IOR metrics might be—-

1. Input metrics
The effort, quantified by engineering-hour, that went into producing the source code modules. The effort is broken down by activity (coding, documenting).

Since the source code module work product's creation depends on the work products that preceded it such as the design specs, the input metrics would also include the Output and Results metrics of those work products.

2. Output Metrics
The size of the source code module as quantified by SLOC (Source Lines Of Code).

The complexity of the source code module as quantified by cyclomatic complexity.

3. Results metrics
The quality of the source code module as quantified by the number of defects and the number of changes made since checking.

The premise of their IOR method is that these metrics on the work products are fundamental and independent of management's particular goals and will therefore provide answers to any relevant questions. The metrics themselves will stimulate questions and provide insight about the software development process.

In the next issue, I will describe some metrics commonly used in industry.

If you missed Part 1 or Part 2. Please call Al Leibee X2-1665, or Jennifer Gibson, X3-8543 for a copy.

Call for Participation
IEEE Software Engineering Standards Safety Planning Group

The "Master Plan for Software Engineering Standards" was approved and published by the IEEE Software Engineering Standards Committee (SESC) in December 1993. This plan "documents a statement of direction for the improvement of software engineering standards for a ten year period." A number of planning groups were established in 1994 to prepare plans on specific topics; more are being created in 1995. The Software Safety Planning Group (SSPG) was created in early May, 1995. I am the chairman of this planning group.

The purpose of the SSPG is "to determine a statement of direction for IEEE standards for software safety." The SSPG is responsible for refining its initial charter and obtaining SESC approval of the revised charter, and preparing a draft Action Plan. Target dates for approval of the revised charter and action plan are September 1995 and June 1996, respectively.

I invite all interested persons to join the planning group. I expect most of the work to be done via electronic mail, so distance and travel difficulties will not preclude participation. I am particularly interested in including people from all parts of the world. Each member is welcome to participate as much or as little as desired - from helping write the action plan to passive observation - all are welcome.

If you wish to join the planning group, please submit this information to address listed below:

Your name, Company affiliation, (if any),
Regular mail address, Phone number,
Fax number, Electronic mail address.

Dr. J. Dennis Lawrence
Lawrence Livermore National Laboratory
7000 East Avenue, L-632
Livermore, CA 94550 USA
E-mail: lawrence2@llnl.gov
j.lawrence@ieee.org

... Continued From page 3

Metrics Food For Thought and Facts

of Rubin's seminars. The survey also showed that the top three information technology priorities for companies are—

1. Business alignment.
2. Reengineering the business with information technology.
3. Upgrading skills.

Within the software producing industry, the top three are Quality, Metrics, and Skills.

Walkthrough Tutorial

The Software Technology Center has a 1.5 hour Walkthrough Tutorial that is designed to be given to project teams.

The tutorial explains the benefits and mechanics of a walkthrough and includes a hands-on session. If your project team is interested in learning more about walkthroughs in a short amount of time, then contact the STC at ext. 3-8333. For information on the effectiveness of software inspections (similar to walkthroughs) in industry, see the related article in the Jan-Feb '95 issue of this newsletter.

Software Engineering Newsletter

Technical Editor: Jeff Young,
(510) 423-8333, L-548, jeffyoung@llnl.gov

Newsletter Compositor & Designer: Jennifer Gibson,
(510) 423-9347, L-307, jlgibson@llnl.gov

General Information or article submission:
(510) 423-8543, stc@llnl.gov

STC WWW Pages
External: http://www.llnl.gov/stc/stc.html
or
Internal: http://www.llnl.gov/llnl_only/stc/